## Московский Авиационный институт (Национальный исследовательский университет)

Факультет №8

# «Информационные технологии и прикладная математика»

**Кафедра 806**

**«Вычислительная математика и программирование»**

**Курсовая работа**

по теме

«Разреженные матрицы»

|  |  |
| --- | --- |
| Студент: | Петров И. О. |
| Группа: | М8О-106Б-21 |
| Преподаватель: | Дубинин А.В. |
| Подпись: |  |
| Оценка: |  |

Москва, 2022

# Задание

Составить программу на языке Си с функциями для обработки прямоугольных разреженных матриц с элементами целого типа, которая:

Вводит матрицы различного размера с одновременным размещением ненулевых элементов в разреженной матрице в соответствии с заданной схемой;

Печатает введенные матрицы во внутреннем представлении и в обычном виде; Выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путем обращения к соответствующим функциям;

Печатает результат преобразования во внутреннем представлении и в обычном виде.

В процедурах и функциях предусмотреть проверки и печать сообщений в случаях ошибок в задании параметров. Для отладки использовать матрицы, содержащие 5- 10% ненулевых элементов, с максимальным числом элементов 100.

**Вариант схемы размещения матрицы:**

1. Цепочка ненулевых элементов в векторе A со строчным индексированием.



**Вариант преобразования:**



# Общий метод решения

Разреженная матрица-это матрица с преимущественно нулевыми элементами. В противном случае, если бо́ льшая часть элементов матрицы ненулевые, матрица считается плотной.

Хранение разреженной матрицы в памяти должно обеспечивать:

1. экономию памяти
2. быстрый доступ к нулевым и ненулевым элементам по их индексу.

Поэтому, хранение разреженный матрицы с помощью трех векторов крайне удобно.

Мы представляем исходную матрицу M ,размерами m x n, cодержащую N ненулевых значений в виде трёх массивов:

* *массив значений* - массив размера N , в котором хранятся ненулевые значения взятые подряд из первой непустой строки, затем идут значения из следующей непустой строки и т.д.
* *массив номеров столбцов*- массив размера N и хранит номера столбцов,

соответствующих элементов из *массива значений.*

* *массив индексации строк n + 1* - массив размера(количество строк), для индекса i хранит количество ненулевых элементов в строках до i – 1 включительно

# Общее сведения о программе

Тестирование, отладка и протоколирование программ производилось на домашнем ПК.

Система программирования: GNU C/C++.

**Структура:**

1.vector\_int - структура вектора чисел,состоит из двух полей типа int, задающих текущее число элементов в векторе (num\_el) и максимальное число элементов (size),которое может содержаться в векторе,а также массив чисел int.

# Алгоритм решения поставленной задачи

Для обработки разреженных матриц опишем структуру вектора с его множеством операций и реализуем вектор на Си. Отдельно опишем функции для обработки разреженных матриц:

1. Считывание матриц в обычном виде из файла с преобразованием в вектор согласно заданной схеме размещения
2. Выполнение заданного преобразования
3. Печать вектора(схемы размещения ненулевых элементов разреженной матрицы)
4. Печать матрицы в естественном виде

# Текст программы

## vector.h

#ifndef \_VECTOR\_H\_ #define \_VECTOR\_H\_

#include <stdio.h> #include <stdlib.h>

typedef struct vector { int size;

int num\_el; int \*elem;

} vector;

void create\_vector(vector \*v); int empty\_vector(vector \*v); int size(vector \*v);

void resize(vector \*v);

void push\_back(vector \*v, int value); void print\_vector(vector \*v);

void load\_matrix(vector \*string, vector \*column, vector \*value);

void create\_unit\_matrix(vector \*string\_u, vector \*column\_u, vector \*value\_u,

vector \*string); void matrix\_mult(vector \*value, int const);

void sum(vector \*string\_u, vector \*column\_u, vector \*value\_u, vector \*string, vector \*column, vector \*value);

void print\_matrix(vector \*string, vector \*column, vector \*value); #endif

## vector.c

#include "vector.h" #include <stdio.h> #include <stdlib.h>

void create\_vector(vector \*v)

{

v->size = 0;

v->num\_el = 0;

v->elem = (int \*) (malloc(sizeof(int) \* v->size));

}

int empty\_vector(vector \*v)

{

return(v->num\_el == 0) ? 1 : 0;

}

int size(vector \*v)

{

return(v->size);

}

void resize(vector \*v)

{

v->size++;

v->elem = (int \*) (realloc(v->elem, sizeof(int) \* v->size));

}

void push\_back(vector \*v, int value)

{

if (v->size == v->num\_el) { resize(v);

}

v->elem[v->num\_el] = value; v->num\_el++;

}

void print\_vector(vector \*v)

{

if (empty\_vector(v)){

printf("Вектор пуст.");

} else {

for (int i = 0; i < v->num\_el; i++) { printf("%d ", v->elem[i]);

}

printf("\n");

}

}

void load\_matrix(vector \*string, vector \*column, vector \*value)

{

char name[20];

printf("введите название файла: "); scanf("%s", name);

FILE \*f;

if ((f = fopen(name, "r")) == NULL) { printf("Ошибка, не удается открыть файл!"); return;

}

int num\_string, num\_column, val, count = 0;

fscanf(f, "%d%d", &num\_string, &num\_column); if (num\_string != num\_column) {

printf("Матрица должна быть размера n\*n, иначе операция невозможна!\n"); return;

}

for (int i = 0; i < num\_string; i++) { push\_back(string, count);

for (int j = 0; j < num\_column; j++) { fscanf(f, "%d", &val);

if (val != 0) {

push\_back(column, j+1); push\_back(value, val); count++;

}

}

}

}

void create\_unit\_matrix(vector \*string\_u, vector \*column\_u, vector \*value\_u,

vector \*string)

{

for (int i = 0; i < string->num\_el; i++) { push\_back(string\_u, i); push\_back(column\_u, i+1);

push\_back(value\_u, 1);

}

}

void matrix\_mult(vector \*value, int c)

{

for (int i = 0; i < value->num\_el; i++) { value->elem[i]\*= c;

}

}

void print\_matrix(vector \*string, vector \*column, vector \*value)

{

int col\_i = 0; int str\_i = 1;

for (int i = 0; i < string->num\_el; i++) {

for (int j = 1; j <= string->num\_el; j++) { if (column->elem[col\_i] == j) {

printf("%d ", value->elem[col\_i]); col\_i++;

} else {

printf("0 ");

}

if (str\_i < string->num\_el) {

if (string->elem[str\_i] == col\_i) { str\_i++;

for (int k = 0; k < (string->num\_el - j); k++) { printf("0 ");

}

printf("\n"); break;

}

}

}

}

printf("\n\n");

}

void sum(vector \*string\_u, vector \*column\_u, vector \*value\_u, vector \*string, vector \*column, vector \*value)

{

int m[100][100];

int col\_i = 0; int str\_i = 1;

for (int i = 0; i < string->num\_el; i++) {

for (int j = 1; j <= string->num\_el; j++) { if (column->elem[col\_i] == j) {

m[i][j-1] = value->elem[col\_i]; col\_i++;

} else {

m[i][j-1] = 0;

}

if (str\_i < string->num\_el) {

if (string->elem[str\_i] == col\_i) { str\_i++;

for (int k = 0; k < (string->num\_el - j); k++) { m[i][j] = 0;

j++;

}

break;

}

}

}

}

for (int i = 0; i < value\_u->num\_el; i++) { m[i][i]+= value\_u->elem[i];

}

for (int i = 0; i < string->num\_el; i++) {

for (int j = 0; j < string->num\_el; j++) { printf("%d ", m[i][j]);

}

printf("\n");

}

}

## main.c

#include "vector.h" #include <stdio.h> #include <stdlib.h>

int main(void)

{

vector string, column, value, string\_u, column\_u, value\_u; int a, b;

create\_vector(&string); create\_vector(&column); create\_vector(&value); load\_matrix(&string, &column, &value); printf("разреженная матрица:\n"); print\_matrix(&string, &column, &value);

printf("введите число, на которое хотите умножить разреженную матрицу: "); scanf("%d", &a);

printf("\n");

printf("разреженная матрица после преобразования:\n"); matrix\_mult(&value, a);

print\_matrix(&string, &column, &value);

create\_vector(&string\_u); create\_vector(&column\_u); create\_vector(&value\_u);

create\_unit\_matrix(&string\_u, &column\_u, &value\_u, &string); printf("единичная матрица:\n");

print\_matrix(&string\_u, &column\_u, &value\_u);

printf("введите число, на которое хотите умножить единичную матрицу: "); scanf("%d", &b);

printf("\n");

printf("единичная матрица после преобразования:\n"); matrix\_mult(&value\_u, b);

print\_matrix(&string\_u, &column\_u, &value\_u);

printf("матрочлен:\n"); sum(&string\_u, &column\_u, &value\_u,

&string, &column, &value);

return 0;

}

# Тестирование

## Входные данные:

4 4

0 0 0 5

0 2 0 0

1 3 0 0

0 0 4 0

## Вывод:

введите название файла: file.txt Vector of strings: 0 1 2 4

Vector of columns: 4 2 1 2 3

Vector of elemeents: 5 2 1 3 4 разреженная матрица:

0 0 0 5

0 2 0 0

1 3 0 0

0 0 4 0

введите число, на которое хотите умножить разреженную матрицу: 2 разреженная матрица после преобразования:

0 0 0 10

0 4 0 0

2 6 0 0

0 0 8 0

единичная матрица:

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

введите число, на которое хотите умножить единичную матрицу: 3 единичная матрица после преобразования:

3 0 0 0

0 3 0 0

0 0 3 0

0 0 0 3

матрочлен: 3 0 0 10

0 7 0 0

2 6 3 0

0 0 8 3

|  |
| --- |
| **Входные данные:**  4 4  0 0 0 5  0 2 0 0  1 3 0 0  0 0 4 0 |
| **Вывод:**  введите название файла: file.txt Vector of strings: 0 1 2 4  Vector of columns: 4 2 1 2 3  Vector of elemeents: 5 2 1 3 4 разреженная матрица:  0 0 0 5  0 2 0 0  1 3 0 0  0 0 4 0  введите число, на которое хотите умножить разреженную матрицу: 0 разреженная матрица после преобразования:  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  единичная матрица:  1 0 0 0  0 1 0 0  0 0 1 0  0 0 0 1  введите число, на которое хотите умножить единичную матрицу: 0 единичная матрица после преобразования:  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  матрочлен: 0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0 |
| **Входные данные:**  3 3  0 0 5  4 1 0 |

|  |
| --- |
| 3 0 0 |
| **Вывод:**  введите название файла: file.txt Vector of strings: 0 1 3  Vector of columns: 3 1 2 1  Vector of elemeents: 5 4 1 3 разреженная матрица:  0 0 5  4 1 0  3 0 0  введите число, на которое хотите умножить разреженную матрицу: -1 разреженная матрица после преобразования:  0 0 -5  -4 -1 0  -3 0 0  единичная матрица:  1 0 0  0 1 0  0 0 1  введите число, на которое хотите умножить единичную матрицу: 1 единичная матрица после преобразования:  1 0 0  0 1 0  0 0 1  матрочлен: 1 0 -5  -4 0 0  -3 0 1 |

# Вывод по задаче

Цель задания достигнута – реализовано сжатое хранение разреженных матриц на основе исходных текстовых файлов. В ходе выполнения работы был составлен код на Си, осуществляющий чтение разреженной матрицы из файла и хранение её в более компактном виде. Получены навыки работы с динамическими структурами данных: векторами, списками, улучшены навыки работы с файлами и ссылками.

Прояснены основные моменты, связанные с адресацией, с использованием файлов. Написание вручную динамических структур помогает приобрести опыт в программировании на языке высокого уровня (конкретно – Си) и лучше понять принципы работы и способы взаимодействия основных единиц данного языка.